

Ayan Palchaudhuri  
Rajat Subhra Chakraborty

# High Performance Integer Arithmetic Circuit Design on FPGA

Architecture, Implementation and  
Design Automation



Springer

# **Springer Series in Advanced Microelectronics**

Volume 51

## **Series editors**

Kukjin Chun, Seoul, Korea, Republic of (South Korea)

Kiyoo Itoh, Tokyo, Japan

Thomas H. Lee, Stanford, CA, USA

Rino Micheloni, Vimercate (MB), Italy

Takayasu Sakurai, Tokyo, Japan

Willy M.C. Sansen, Leuven, Belgium

Doris Schmitt-Landsiedel, München, Germany

The Springer Series in Advanced Microelectronics provides systematic information on all the topics relevant for the design, processing, and manufacturing of microelectronic devices. The books, each prepared by leading researchers or engineers in their fields, cover the basic and advanced aspects of topics such as wafer processing, materials, device design, device technologies, circuit design, VLSI implementation, and subsystem technology. The series forms a bridge between physics and engineering and the volumes will appeal to practicing engineers as well as research scientists.

More information about this series at <http://www.springer.com/series/4076>

Ayan Palchaudhuri · Rajat Subhra Chakraborty

# High Performance Integer Arithmetic Circuit Design on FPGA

Architecture, Implementation and Design  
Automation

Ayan Palchaudhuri  
Department of Electronics and Electrical  
Communication Engineering  
Indian Institute of Technology Kharagpur  
Kharagpur, West Bengal  
India

Rajat Subhra Chakraborty  
Department of Computer Science  
and Engineering  
Indian Institute of Technology Kharagpur  
Kharagpur, West Bengal  
India

ISSN 1437-0387                      ISSN 2197-6643 (electronic)  
Springer Series in Advanced Microelectronics  
ISBN 978-81-322-2519-5              ISBN 978-81-322-2520-1 (eBook)  
DOI 10.1007/978-81-322-2520-1

Library of Congress Control Number: 2015943832

Springer New Delhi Heidelberg New York Dordrecht London  
© Springer India 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer (India) Pvt. Ltd. is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

*To Ma, for her continuous prayers,  
support and inspiration.*

Ayan Palchaudhuri

*To Bunto, for her patience, love  
and understanding.*

Rajat Subhra Chakraborty

# Preface

Realization of high performance arithmetic circuits targeted towards a specific family of the high-end Field Programmable Gate Arrays (FPGAs) continue to remain a challenging problem. Many fast arithmetic circuits proposed over the decades may not be amenable to efficient realization on a selected FPGA architecture. Experience has shown that current CAD tools for FPGAs are often unable to infer the native architectural components efficiently from the given input Hardware Description Language (HDL) specification of the circuit, as they explore only a small design space close to the input architectural description. The logic synthesis techniques inherent to the CAD tools are also often unable to apply the proper Boolean identities and perform appropriate algebraic factoring and sub-expression sharing, especially when intermediate signals are tapped out or registered. *Primitive instantiation* is an effective approach for optimization of designs on the Xilinx FPGA platform, and is often simpler than rewriting the Register Transfer Level (RTL) code to coax the logic synthesis tool to infer the desired architectural components. In addition, the FPGA CAD tools often fail to achieve an efficient placement of logic blocks on the FPGA fabric, resulting in higher routing delays.

In this book, we describe the optimized implementations of several arithmetic datapath, controlpath, and pseudorandom sequence generator circuits. We explore regular, modular, cascadable, and bit-sliced architectures for these circuits, by directly instantiating the target FPGA-specific primitives in the HDL specifications of the circuits. We justify every proposed architecture with detailed mathematical analyses. We improve performance by enforcing a constrained placement of the circuit building blocks, by placing the logically related hardware primitives in close proximity to one another, thereby minimizing the routing delay. This is accomplished by supplying relevant placement constraints in the Xilinx proprietary “User Constraints File” (.ucf) format to the FPGA CAD tool.

Taking advantage of the regularity of the architectures of the circuits proposed by us, the HDL specifications of the circuits as well as the placement constraints can be automatically generated. We have implemented a GUI-based CAD tool named *FlexiCore* integrated with the *Xilinx ISE* (Integrated Software Environment)

design environment for design automation of platform-specific high performance arithmetic circuits from user-level specifications. This tool was used to implement the proposed circuits, as well as hardware implementations of two integer arithmetic algorithms (Greatest Common Divisor (GCD) using Binary GCD algorithm and matrix multiplication using Distributed Arithmetic (DA)) where several of the proposed circuits were used as building blocks. Implementation results demonstrate higher performance and superior operand-width scalability at acceptable power-delay product (PDP) for the proposed circuits, with respect to implementations derived through other existing approaches.

Kharagpur

Ayan Palchaudhuri  
Rajat Subhra Chakraborty

# Acknowledgments

The authors thank Prof. Anindya Sundar Dhar, Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, and Dr. Debdeep Mukhopadhyay, Department of Computer Science and Engineering, IIT Kharagpur, for their valuable insights into the work. The authors also acknowledge two undergraduate students of the Department of Computer Science and Engineering, IIT Kharagpur, Mohammad Salman and Sreemukh Kardas, for their contributions in developing the proposed CAD tool, *FlexiCore*.

This research was funded by a start-up research grant provided by IIT Kharagpur to Rajat Subhra Chakraborty.

# Contents

|          |                                                                                                      |           |
|----------|------------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                                  | <b>1</b>  |
| 1.1      | Background of FPGA-Based Design                                                                      | 1         |
| 1.2      | Limitations of FPGA CAD Tools                                                                        | 2         |
| 1.3      | Overview of Design Philosophy for FPGAs                                                              | 3         |
| 1.3.1    | Target FPGA-Specific Hardware Primitive<br>Instantiation                                             | 3         |
| 1.4      | Existing FPGA CAD Tools                                                                              | 4         |
| 1.4.1    | Xilinx IP Core Generator                                                                             | 4         |
| 1.4.2    | <i>FloPoCo</i> (Floating-Point Cores)                                                                | 5         |
| 1.5      | Recent Works on High Performance Circuit Realization<br>on Xilinx FPGAs                              | 6         |
| 1.6      | Major Contributions of the Book                                                                      | 6         |
| 1.7      | Organization of the Book                                                                             | 8         |
| 1.8      | Summary                                                                                              | 9         |
|          | References                                                                                           | 9         |
| <b>2</b> | <b>Architecture of Target FPGA Platform</b>                                                          | <b>11</b> |
| 2.1      | Introduction                                                                                         | 11        |
| 2.2      | Fabric Slice Architecture for Virtex-5 FPGAs                                                         | 12        |
| 2.3      | Fabric Slice Architecture for Virtex-6 FPGAs                                                         | 14        |
| 2.4      | DSP Slice Architecture for Virtex-5 and Virtex-6 FPGAs                                               | 15        |
| 2.5      | Implementation Overview                                                                              | 16        |
| 2.6      | Summary                                                                                              | 17        |
|          | References                                                                                           | 17        |
| <b>3</b> | <b>A Fabric Component Based Design Approach<br/>for High-Performance Integer Arithmetic Circuits</b> | <b>19</b> |
| 3.1      | Introduction                                                                                         | 19        |
| 3.2      | Existing Work                                                                                        | 20        |

|          |                                                                |           |
|----------|----------------------------------------------------------------|-----------|
| 3.3      | Guidelines for High-Performance Realization . . . . .          | 21        |
| 3.4      | Summary . . . . .                                              | 28        |
|          | References . . . . .                                           | 28        |
| <b>4</b> | <b>Architecture of Datapath Circuits . . . . .</b>             | <b>31</b> |
| 4.1      | Introduction . . . . .                                         | 31        |
| 4.2      | Integer Adder/Subtractor Architecture . . . . .                | 32        |
| 4.2.1    | Hybrid Ripple Carry Adder (Hybrid RCA) . . . . .               | 32        |
| 4.2.2    | Xilinx DSP Slice-Based Adder . . . . .                         | 34        |
| 4.2.3    | FloPoCo-Based Adder . . . . .                                  | 35        |
| 4.2.4    | Fast Carry Adder Using Carry-Lookahead Mechanism . . . . .     | 35        |
| 4.2.5    | Adder Implementation Results . . . . .                         | 39        |
| 4.3      | Absolute Difference Circuit Architecture . . . . .             | 43        |
| 4.3.1    | Proposed Absolute Difference Circuit . . . . .                 | 43        |
| 4.3.2    | DSP Slice-Based Absolute Difference Circuit . . . . .          | 44        |
| 4.3.3    | FloPoCo-Based Absolute Difference Circuit . . . . .            | 45        |
| 4.3.4    | Absolute Difference Circuit Implementation Results . . . . .   | 46        |
| 4.4      | Integer Multiplier Architecture . . . . .                      | 48        |
| 4.4.1    | Unsigned Integer Multiplier . . . . .                          | 48        |
| 4.4.2    | Two's Complement Multiplier . . . . .                          | 49        |
| 4.4.3    | Combined Unsigned and Two's Complement<br>Multiplier . . . . . | 51        |
| 4.4.4    | DSP Slice-Based Signed Multiplier . . . . .                    | 55        |
| 4.4.5    | FloPoCo-Based Signed Multiplier . . . . .                      | 55        |
| 4.4.6    | Multiplier Implementation Results . . . . .                    | 55        |
| 4.5      | Integer Squarer Architecture . . . . .                         | 57        |
| 4.5.1    | Unsigned Squarers . . . . .                                    | 59        |
| 4.5.2    | Two's Complement Squarers . . . . .                            | 60        |
| 4.5.3    | Combined Unsigned and Two's Complement Squarer . . . . .       | 64        |
| 4.5.4    | DSP Slice-Based Squarers . . . . .                             | 65        |
| 4.5.5    | FloPoCo-Based Squarers . . . . .                               | 67        |
| 4.5.6    | Squarer Implementation Results . . . . .                       | 67        |
| 4.6      | Universal Shift Register Architecture . . . . .                | 67        |
| 4.6.1    | Universal Shift Register . . . . .                             | 67        |
| 4.6.2    | Universal Shift Register Implementation Results . . . . .      | 69        |
| 4.7      | Summary . . . . .                                              | 70        |
|          | References . . . . .                                           | 71        |
| <b>5</b> | <b>Architecture of Controlpath Circuits . . . . .</b>          | <b>73</b> |
| 5.1      | Introduction . . . . .                                         | 73        |
| 5.2      | Integer Comparator Architecture . . . . .                      | 73        |
| 5.2.1    | Proposed Comparator Architecture . . . . .                     | 73        |
| 5.2.2    | DSP Slice-Based Comparator . . . . .                           | 76        |
| 5.2.3    | Comparator Implementation Results . . . . .                    | 77        |

|          |                                                                         |            |
|----------|-------------------------------------------------------------------------|------------|
| 5.3      | Loadable Bidirectional Binary Counter Architecture . . . . .            | 79         |
| 5.3.1    | Proposed Counter Architecture . . . . .                                 | 79         |
| 5.3.2    | DSP Slice-Based Counter . . . . .                                       | 81         |
| 5.3.3    | Counter Implementation Results. . . . .                                 | 81         |
| 5.4      | Summary . . . . .                                                       | 82         |
|          | References . . . . .                                                    | 83         |
| <b>6</b> | <b>Compact FPGA Implementation of Linear Cellular Automata . . . .</b>  | <b>85</b>  |
| 6.1      | Introduction. . . . .                                                   | 85         |
| 6.2      | Preliminaries on Cellular Automata . . . . .                            | 86         |
| 6.3      | Adapting CA to the Native FPGA Architecture . . . . .                   | 88         |
| 6.4      | CA Implementation Results . . . . .                                     | 89         |
| 6.5      | Summary . . . . .                                                       | 90         |
|          | References . . . . .                                                    | 91         |
| <b>7</b> | <b>Design Automation and Case Studies . . . . .</b>                     | <b>93</b>  |
| 7.1      | Introduction. . . . .                                                   | 93         |
| 7.2      | The FlexiCore CAD Tool . . . . .                                        | 94         |
| 7.3      | Case Studies . . . . .                                                  | 97         |
| 7.3.1    | GCD Calculator Circuit . . . . .                                        | 98         |
| 7.3.2    | Distributed Arithmetic-Based Matrix<br>Multiplication Circuit . . . . . | 102        |
| 7.4      | Summary . . . . .                                                       | 106        |
|          | References . . . . .                                                    | 107        |
| <b>8</b> | <b>Conclusions and Future Work . . . . .</b>                            | <b>109</b> |
| 8.1      | Introduction. . . . .                                                   | 109        |
| 8.2      | Contributions of the Book. . . . .                                      | 109        |
| 8.3      | Future Research Directions . . . . .                                    | 110        |
|          | References . . . . .                                                    | 112        |
|          | <b>Index . . . . .</b>                                                  | <b>113</b> |

# Acronyms

|                |                                         |
|----------------|-----------------------------------------|
| ASIC           | Application-Specific Integrated Circuit |
| BIST           | Built-In Self-Test                      |
| CA             | Cellular Automata                       |
| CAA            | Cellular Automata Array                 |
| CAD            | Computer Aided Design                   |
| CDI            | Configuration Data In                   |
| CDO            | Configuration Data Out                  |
| CE             | Clock Enable                            |
| CL             | Combinational Logic                     |
| CLB            | Configurable Logic Block                |
| CORDIC         | COordinate Rotation DIgital Computer    |
| DA             | Distributed Arithmetic                  |
| DFT            | Discrete Fourier Transform              |
| DSP            | Digital Signal Processing               |
| <i>FloPoCo</i> | Floating-Point Cores                    |
| FF             | Flip-Flop                               |
| FFT            | Fast Fourier Transform                  |
| FIR            | Finite Impulse Response                 |
| FPGA           | Field Programmable Gate Array           |
| FSM            | Finite State Machine                    |
| GCD            | Greatest Common Divisor                 |
| GUI            | Graphical User Interface                |
| HDL            | Hardware Description Language           |
| HKMG           | High-K Metal Gate                       |
| HPL            | High Performance Low Power              |
| IP             | Intellectual Property                   |
| ISE            | Integrated Software Environment         |
| LFSR           | Linear Feedback Shift Register          |
| LNS            | Logarithmic Number System               |
| LUT            | Look-Up Table                           |
| MACC           | Multiply Accumulate                     |